Flask Blog 期末報告

學生：陳聖允 5B0G0011
任課老師：李宗儒

背景概述

Flask Blog 這專案是作者 Corey Schafer 在展示如何使用 Flask 框架開發部落格系統的範例專案。此專案旨在教導開發者如何構建一個功能完善的網頁應用程式，並涵蓋多項關鍵技術，包括用戶管理、貼文管理及錯誤處理功能。

透過這個專案，作者希望讓學習者掌握 Flask 框架的基本概念與實際應用，強調 MVC 架構設計的重要性及如何擴展與維護應用程式。此外，此專案還展示了如何安全地處理表單輸入、管理資料庫與建立互動式網頁模板。

目前進度

依照教學影片的進度是做到了 Part 12 的部分，下是已完成的內容與主要功能：

01-Getting Started
設置 Flask 開發環境並建立基本的 Flask 應用程式。

02-Templates
使用 Jinja2 模板語言建立動態 HTML 頁面，並整合主版面設計。

03-Forms and Validation
實現表單驗證，包含註冊與登入功能的表單處理與錯誤提示。

04-Database
使用 SQLAlchemy 設計與初始化資料庫，建立用戶與貼文模型。

05-Package Structure
重構專案目錄結構，將功能模組化並實現 Blueprint。

06-Login Auth
設置用戶登入與登出功能，並透過 Flask-Login 管理用戶狀態。

07-User Account Profile Pic
支援使用者上傳與管理個人頭像及帳號資料更新功能。

08-Posts
建立貼文管理系統，包含新增、編輯與刪除功能。

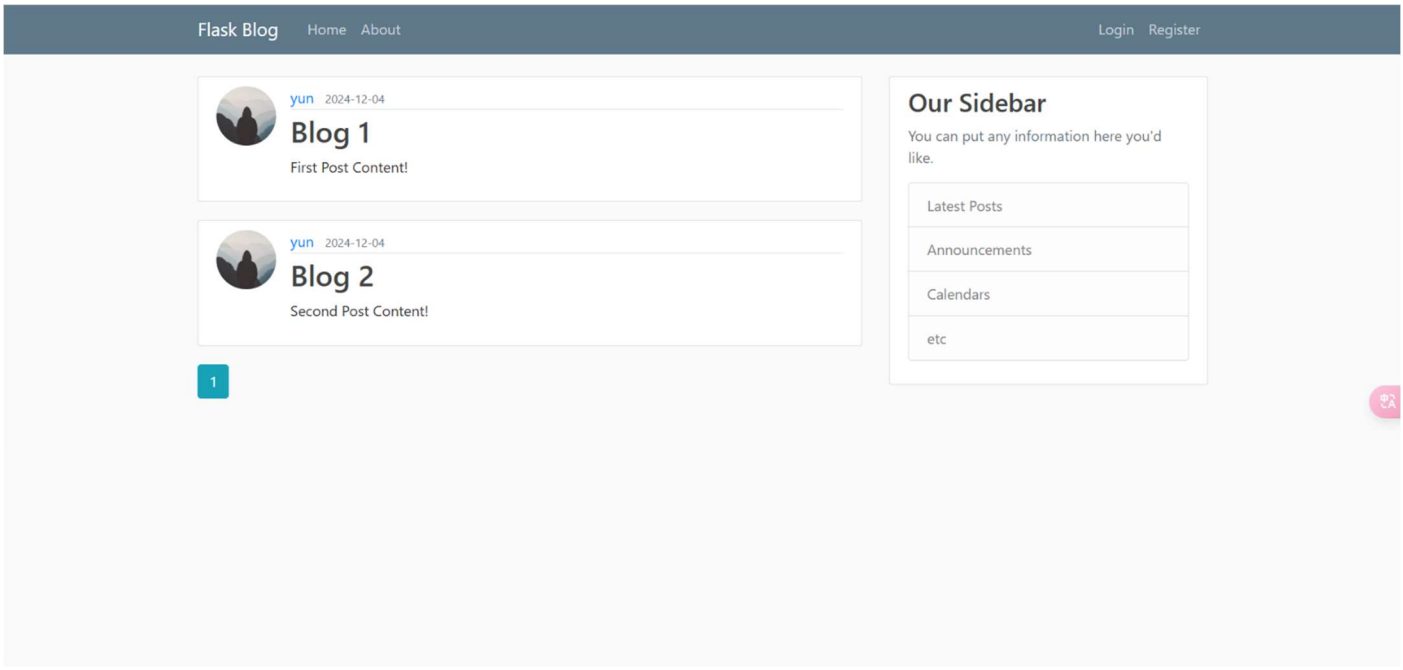
09-Pagination
增加貼文分頁功能，提高瀏覽效率。

10-Password Reset Email
支援密碼重設功能，透過電子郵件發送重設連結。

11-Blueprints
使用 Blueprint 將應用邏輯模組化，提升擴展性與可維護性。

12-Error Pages
設置自訂錯誤頁面 (403、404、500)，增強錯誤處理與友好度。

網頁功能說明



圖一.Flask Blog 首頁

圖一我們可以看到這是部落格的首頁，以下是首頁的功能

1. 頁面標題與導航列

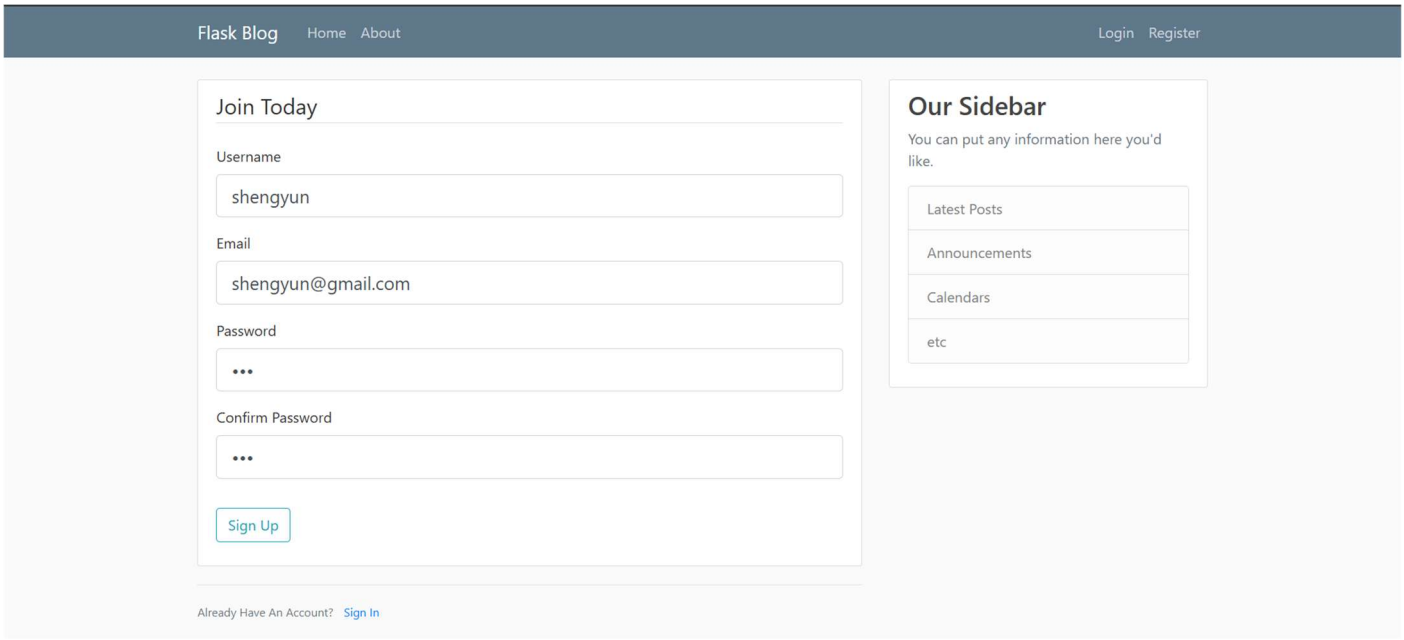
- **標題：** 左上方顯示網站名稱 "Flask Blog"。
- **導航列：** 提供首頁 (Home) 與關於頁面 (About) 的快速連結。
- **登入與註冊：** 右上角顯示登入 (Login) 和註冊 (Register) 按鈕，方便用戶進行帳號操作。
-

2. 主內容區域

- **貼文列表：**
  - 每個貼文包含以下內容：
    1. **作者名稱 (yun)** - 點擊後可以檢視該作者的所有文章。
    2. **發佈日期 (2024-12-04)** - 顯示文章的發布時間。
    3. **標題 (Blog 1、Blog 2)** - 突顯貼文標題。
    4. **內容預覽 (First Post Content!)** - 提供貼文摘要。
  - 貼文顯示在卡片樣式中，提升閱讀體驗。

3. 側邊欄 (Sidebar)

- **標題：** 顯示「Our Sidebar」。
- **資訊區塊：**
  - 可用於展示最新文章、公告、行事曆等內容。
  - 此範例中展示的是模擬功能，包括「Latest Posts」、「Announcements」、「Calendars」與「etc」



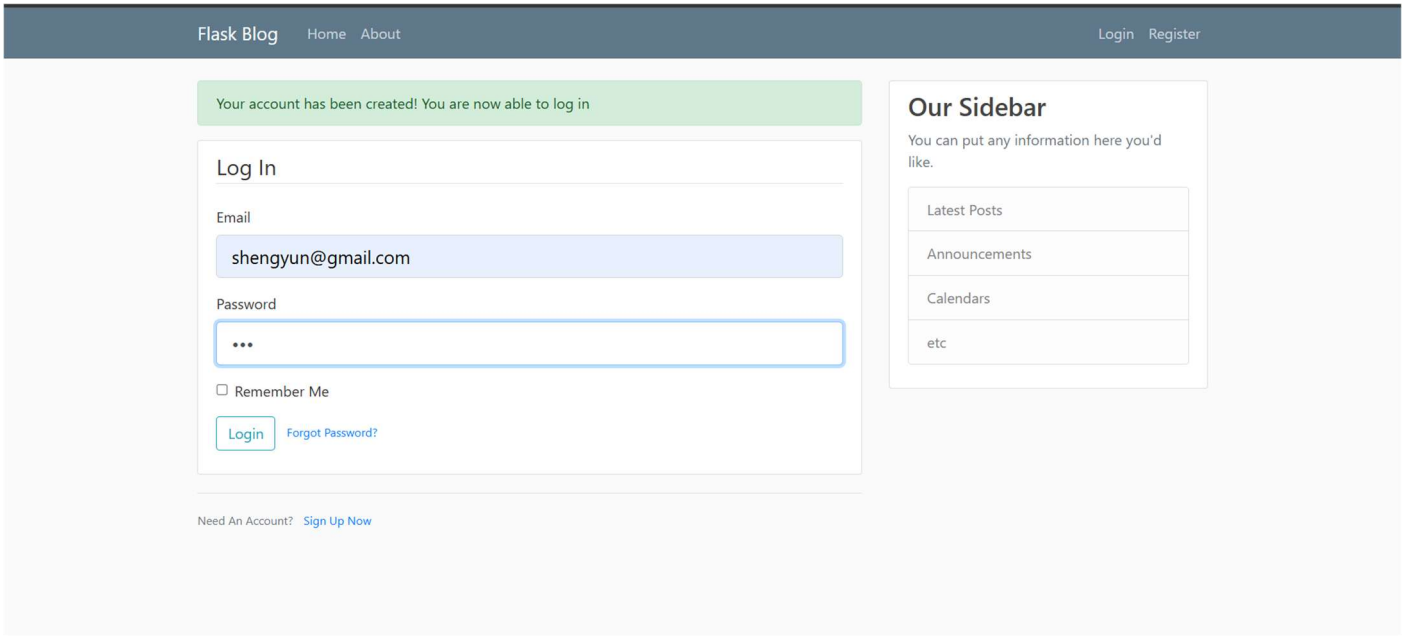
圖二.註冊頁面

從圖一我們可以看到導覽列的右邊有”Login”，”Register”，當我們按下”Register”時就會進入到註冊的畫面了。因為導覽列跟側邊欄都是重複的所以就不再介紹了，Register 的頁面功能有

註冊表單區域

- **標題：** "Join Today" (立即加入)。
- **輸入欄位：**
    1. **Username：** 使用者名稱輸入框。
    2. **Email：** 電子郵件輸入框。
    3. **Password：** 密碼輸入框 (隱藏輸入)。
    4. **Confirm Password：** 確認密碼輸入框，必須與密碼一致。
- **按鈕：**
    - **Sign Up：** 提交表單以完成註冊。
    - **Sign In：** 若已有帳號，可點擊底部連結切換至登入頁面。

此頁面提供使用者註冊帳號的功能，確保資料經過驗證後才能提交。它也支援密碼確認與錯誤提示，確保用戶體驗與安全性。



圖三.Login 頁面

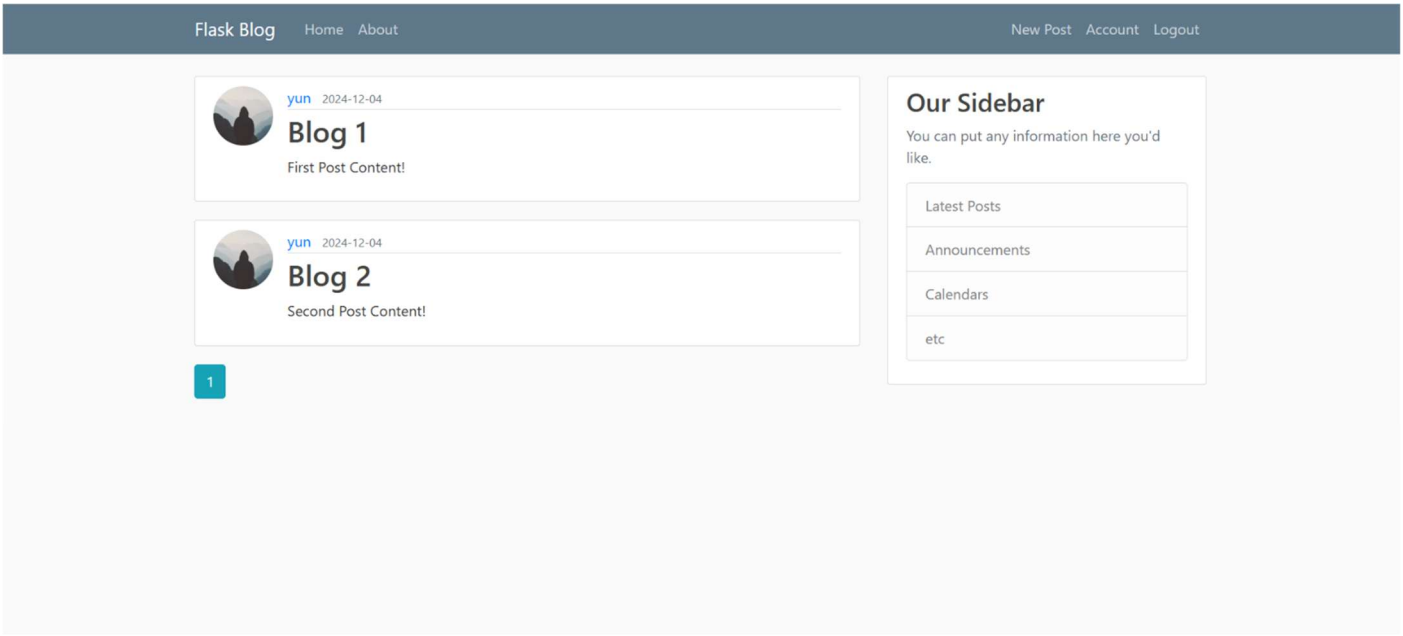
當我們註冊完帳號後看到頂端顯示綠色通知，表示帳戶已成功註冊，現在可以使用憑證進行登入。

:「Your account has been created! You are now able to log in」

登入表單區域

- **標題：** "Log In"。
- **輸入欄位：**
  1. **Email：** 用戶輸入電子郵件地址。
  2. **Password：** 用戶輸入密碼 (隱藏輸入)。
- **選項：**
  - **Remember Me：** 記住登入狀態選項，方便未來快速登入。
- **按鈕：**
  1. **Login：** 提交表單以完成登入。
  2. **Forgot Password?：** 提供忘記密碼的連結，協助用戶重設密碼。

此頁面提供帳號登入功能，支援記住登入狀態與密碼重設功能。透過表單驗證確保輸入資料正確性，並提升使用者操作便利性。



圖四.Login 後的首頁

由圖四看到以下是登入後頁面與登入前頁面的主要差異：

1. 導覽列變化

- **登入前：**
  - 導覽列右上角顯示「Login」與「Register」按鈕，提供登入與註冊功能。
- **登入後：**
  - 導覽列右上角變更為「New Post」、「Account」與「Logout」按鈕：
    1. **New Post：** 可以新增貼文。
    2. **Account：** 進入帳戶管理頁面，更新個人資訊或更換頭像。
    3. **Logout：** 登出帳號並回到登入狀態。

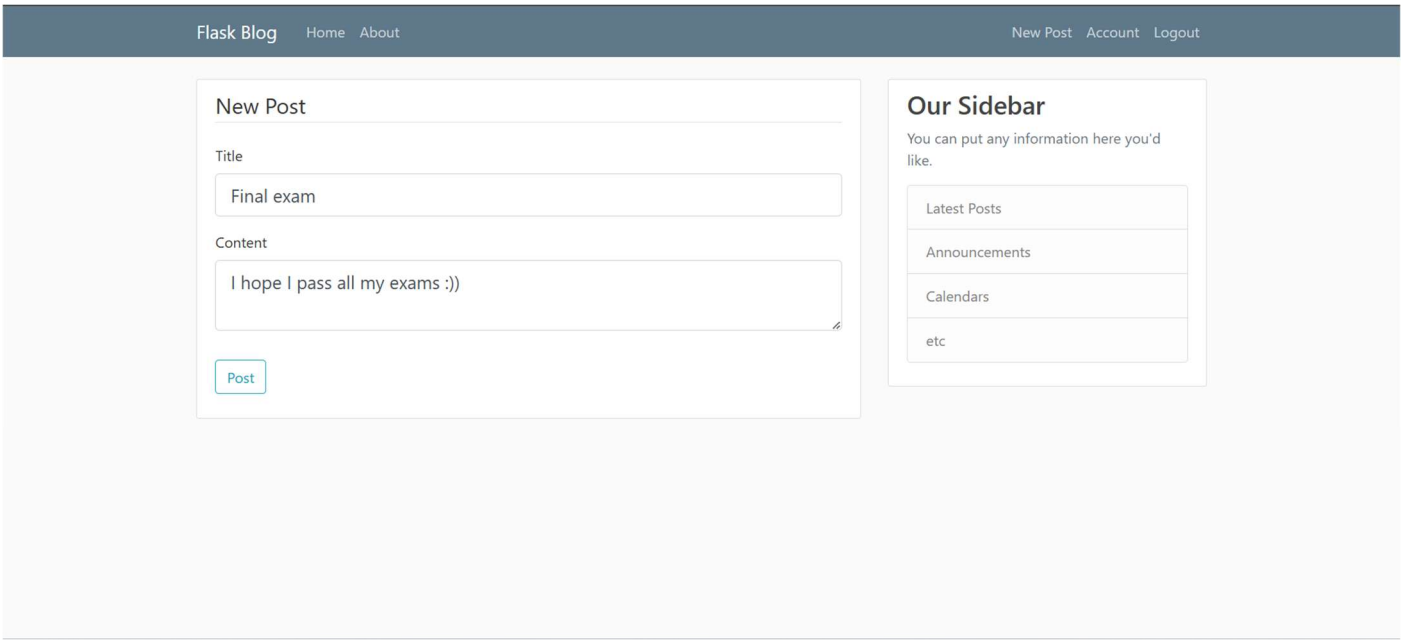
2. 使用者互動功能

- **登入前：**
  - 僅能瀏覽文章與側邊欄的資訊，無法建立或管理貼文。
- **登入後：**
  - 可以新增貼文、管理個人帳號資訊，以及執行登出操作。

3. 功能權限變化

- **登入前：**
  - 只能檢視貼文內容，無法新增、編輯或刪除貼文。
- **登入後：**
  - 新增「New Post」按鈕，支援發佈新貼文與編輯管理文章。

登入後的頁面增加了用戶專屬功能，如新增貼文、帳號管理與登出選項，並顯示用戶名稱與互動按鈕，提供更完整的內容管理與操作權限。



圖五.New Post Page

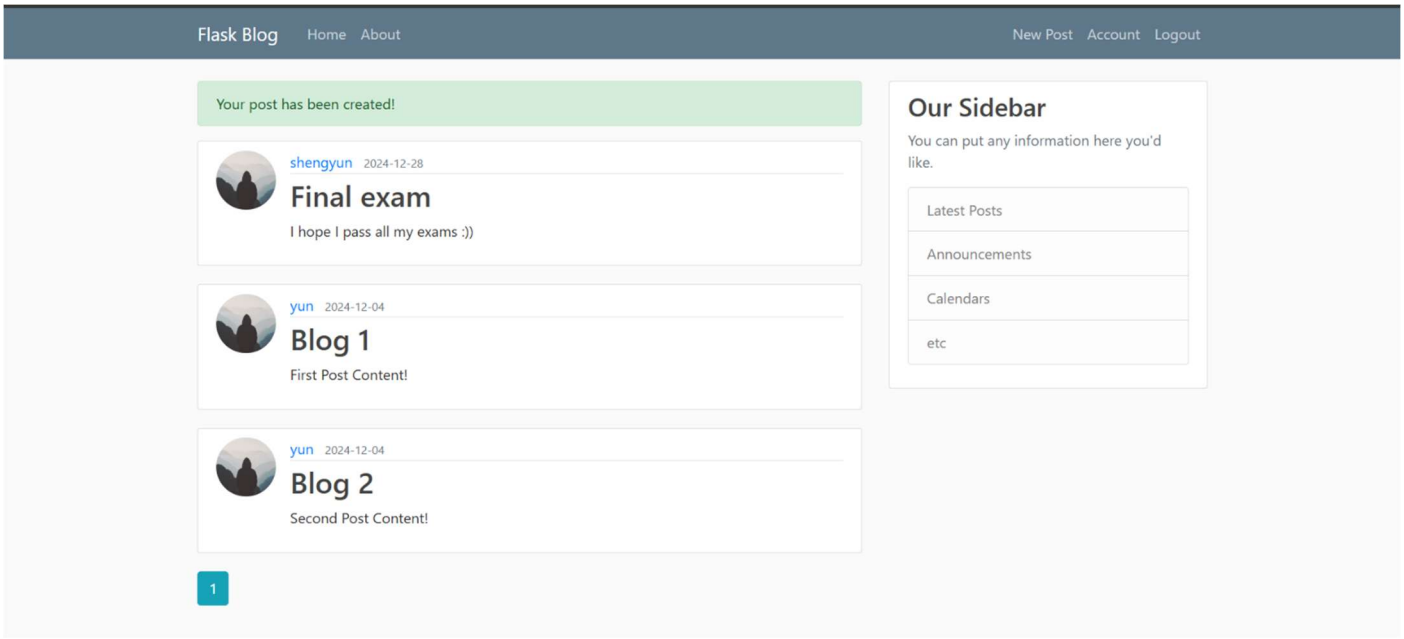
圖五展示了新增貼文(New Post) 頁面。以下是功能說明：

1. 貼文建立區域

- **標題 (Title)**：
  - 使用者可輸入貼文標題 (例如「Final exam」)。
- **內容 (Content)**：
  - 使用者可輸入貼文內容 (例如「I hope I pass all my exams :))」)。
- **提交按鈕 (Post)**：
  - 按下按鈕後將貼文內容發佈到首頁並顯示。

此頁面提供用戶建立新貼文的功能，輸入標題與內容後即可發佈到首頁並顯示。

- 需要登入後才能使用此功能，以保證貼文的作者身份與管理權限。
- 支援表單驗證，確保標題與內容為必填項目。



圖六.Send post 後的狀態

圖六展示 Flask Blog 的首頁，顯示用戶新增貼文後的狀態：

1. 通知欄 (Post Created Message)

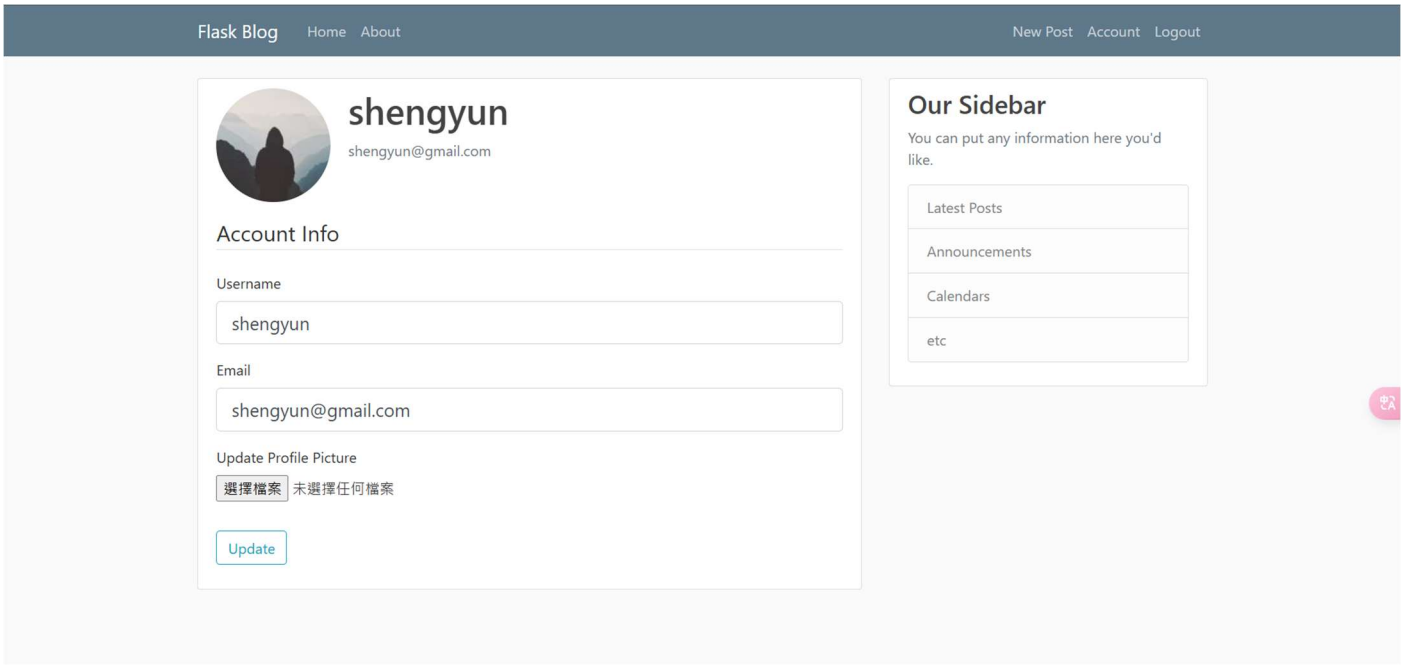
- 頂部綠色通知訊息：「Your post has been created!」
- 表示貼文已成功新增並發佈到首頁。

2. 新增貼文內容展示

- **作者：** "shengyun" 是已登入的用戶名稱，點擊名稱可檢視該用戶的貼文列表。
- **標題：** "Final exam" (貼文標題)。
- **內容：** "I hope I pass all my exams :))" (貼文內容)。
- **日期：** 2024-12-28，顯示貼文發佈的時間。

3. 現有貼文顯示

- **Blog 1 與 Blog 2：** 為之前發佈的貼文，顯示標題、內容與日期。
- **作者：** 兩篇貼文均由 "yun" 發佈，可透過點擊名稱檢視該作者的所有貼文。



圖七.Account Page

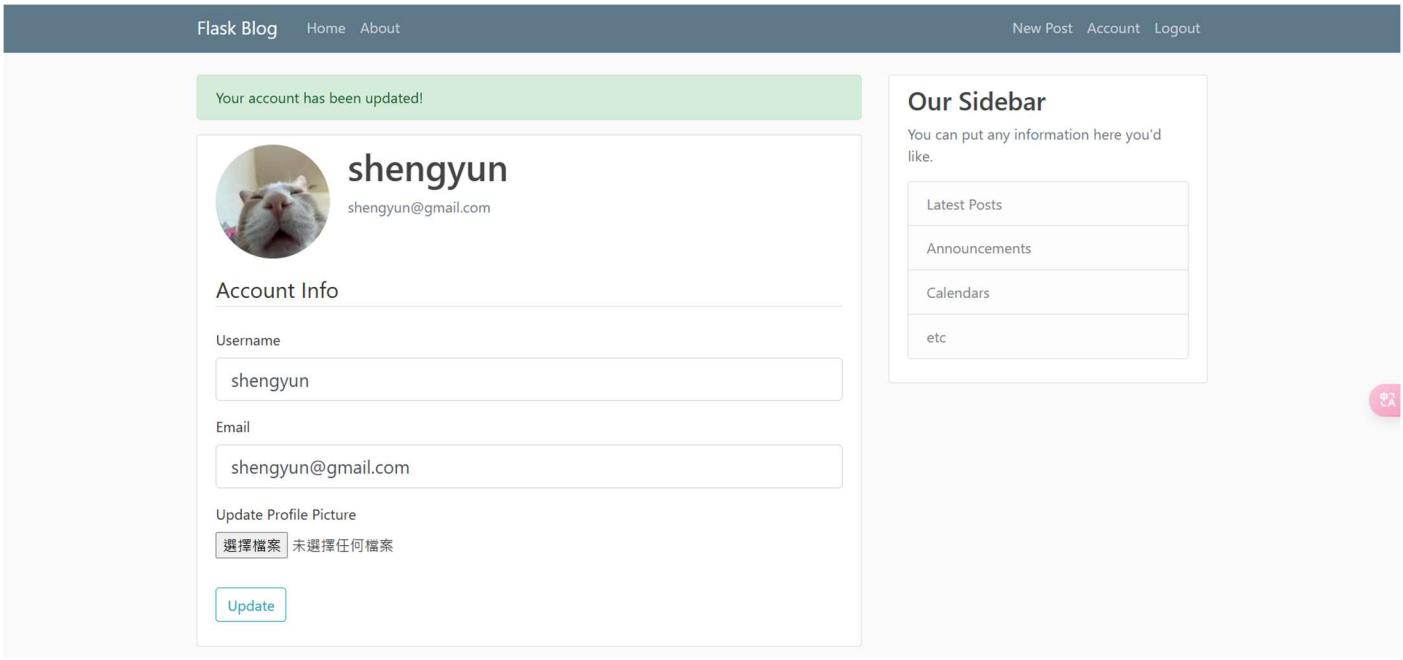
圖七展示 Flask Blog 的帳戶設定頁面 (Account Page)：

1. 個人資料資訊顯示

- **頭像：**
  - 預設頭像顯示在頁面頂部。
  - 用戶可自訂頭像圖片作為個人標誌。
- **帳號資訊：**
  - **Username：** 顯示當前用戶名稱 (shengyun)。
  - **Email：** 顯示當前電子郵件地址 (shengyun@gmail.com)。
- **檔案上傳 (Update Profile Picture)：**
  - 支援上傳新頭像圖片，更新個人資料圖片。
  - 按下「Update」按鈕保存更改。

此頁面提供使用者管理個人資料與頭像的功能，包括：

- 修改帳號名稱與電子郵件地址。
- 上傳個人頭像，更新圖片顯示。
- 支援即時資料驗證與圖片格式限制，提升安全性與使用者體驗。



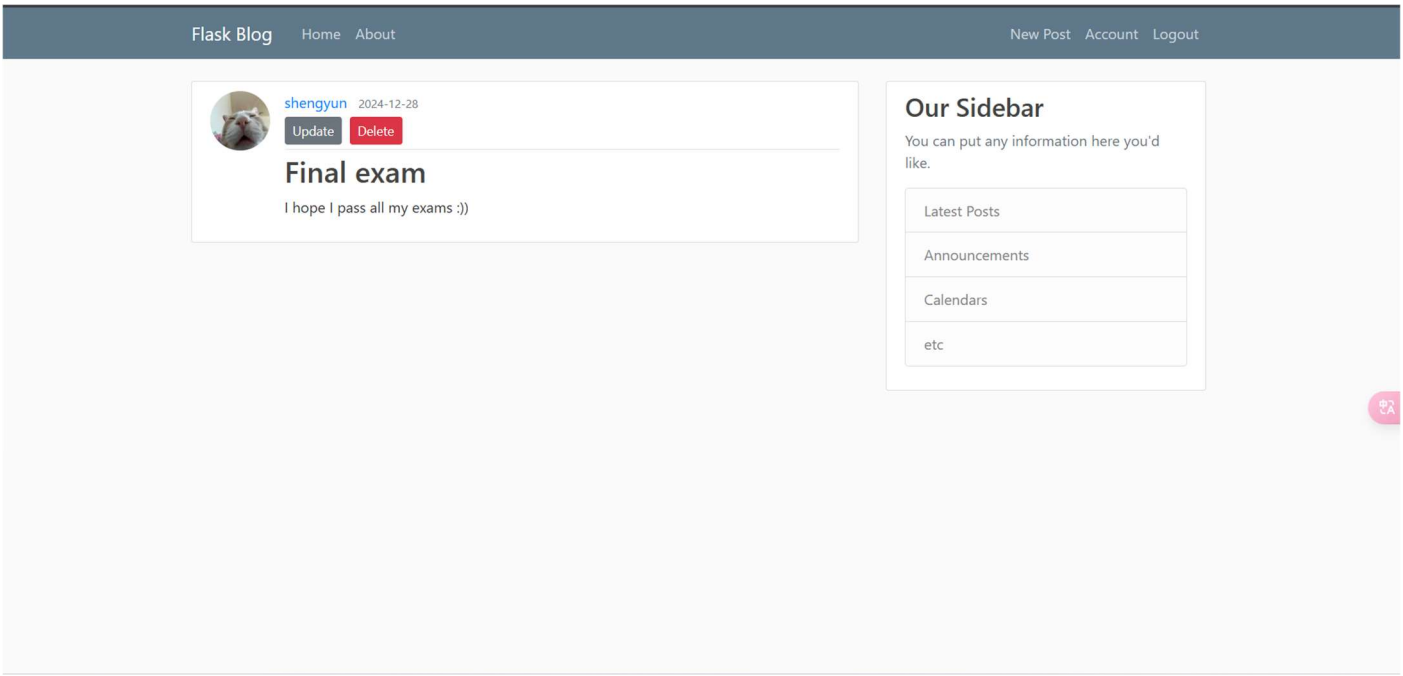
圖八.update Account Page

以下是帳戶更新後的變化：

1. 通知訊息變化
   - 更新前： 無通知訊息或顯示舊資料。
   - 更新後：
     - 頂部新增綠色提示訊息：「Your account has been updated!」
     - 表示帳戶資料或頭像已成功更新並儲存。

2. 頭像變化
   - 更新前： 預設頭像或舊圖片顯示。
   - 更新後：
     - 頭像已變更為使用者新上傳的圖片（例如貓的圖片）。
     - 確保新圖片已儲存在系統內，並透過 Flask 的圖片處理功能進行調整與縮放。

3. 資料更新狀態
   - 更新前： 使用者名稱和電子郵件顯示舊資料。
   - 更新後：
     - 使用者名稱和電子郵件維持不變（若未更新則保持原值）。
     - 提供檢查與確認新資料是否已套用。



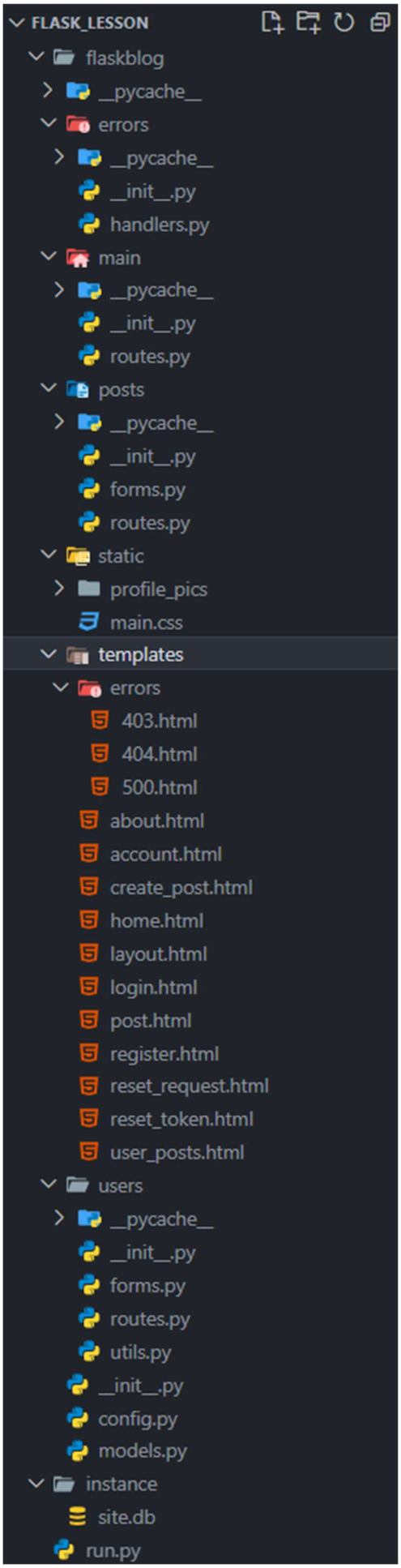
圖九.貼文管理頁面

這張圖片展示 Flask Blog 的貼文管理頁面 (Post Management Page)。

1. 貼文內容顯示
   - 標題： "Final exam" (貼文標題)。
   - 內容： "I hope I pass all my exams :))" (貼文內容)。
   - 作者： "shengyun"，顯示使用者名稱並可點擊連結檢視作者的所有貼文。
   - 發佈日期： 2024-12-28，顯示貼文發佈的日期與時間。

2. 貼文操作按鈕
   - Update 按鈕 (灰色)：
     - 點擊後可進入編輯頁面，更新貼文標題與內容。
   - Delete 按鈕 (紅色)：
     - 點擊後刪除貼文，並返回首頁。
     - 包含刪除確認提示，以避免誤刪資料。

專案結構


FLASK_LESSON
flaskblog
__pycache__
errors
__pycache__
__init__.py
handlers.py
main
__pycache__
__init__.py
routes.py
posts
__pycache__
__init__.py
forms.py
routes.py
static
profile_pics
main.css
templates
errors
403.html
404.html
500.html
about.html
account.html
create_post.html
home.html
layout.html
login.html
post.html
register.html
reset_request.html
reset_token.html
user_posts.html
users
__pycache__
__init__.py
forms.py
routes.py
utils.py
__init__.py
config.py
models.py
instance
site.db
run.py

程式說明

第 1 部分：Getting Started

功能：

- 安裝 Flask 與環境設定。
- 建立 Flask 應用程式並測試啟動。

程式碼分析：

run.py - 啟動 Flask 應用程式

```python
from flaskblog import create_app

# 創建 Flask 應用實例
app = create_app()

# 啟動應用程式，啟用除錯模式
if __name__ == '__main__':
    app.run(debug=True)

```

- 啟動 Flask 應用程式並設定 Debug 模式，便於開發與錯誤追蹤。

第 2 部分：Templates

功能：

- 設計 HTML 模板，使用 Jinja2 動態渲染資料。
- 建立基礎佈局，支援頁面內容嵌入。

程式碼分析：

templates/layout.html - 基本模板設計

```html
<!DOCTYPE html>
<html>
<head>
    <title>Flask Blog</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='main.css') }}">
</head>
<body>
    <nav>
        <a href="{{ url_for('main.home') }}">Home</a>
        <a href="{{ url_for('main.about') }}">About</a>
        {% if current_user.is_authenticated %}
            <a href="{{ url_for('users.account') }}">Account</a>
            <a href="{{ url_for('posts.new_post') }}">New Post</a>
            <a href="{{ url_for('users.logout') }}">Logout</a>
        {% else %}
            <a href="{{ url_for('users.login') }}">Login</a>
            <a href="{{ url_for('users.register') }}">Register</a>
        {% endif %}
    </nav>
    <div>
        {% block content %}{% endblock %}
    </div>
</body>
</html>
```

- 支援動態內容渲染與統一的佈局模板。

使用 `url_for()` 生成路由連結，確保頁面導航功能。

`current_user` 用於檢查登入狀態，控制顯示登入/登出按鈕

第 3 部分：Forms and Validation

功能：

- 使用 Flask-WTF 建立表單與資料驗證。
- 防止 CSRF 攻擊與輸入錯誤。

程式碼分析：

users/forms.py - 註冊表單：

```python
# 使用者註冊表單
class RegistrationForm(FlaskForm):
    # 使用者名稱欄位，必填且長度限制為 2 到 20 個字符
    username = StringField('Username',
                           validators=[DataRequired(), Length(min=2, max=20)])
    # 電子郵件欄位，必填且必須符合電子郵件格式
    email = StringField('Email',
                        validators=[DataRequired(), Email()])
    # 密碼欄位，必填
    password = PasswordField('Password', validators=[DataRequired()])
    # 確認密碼欄位，必填且需與密碼欄位一致
    confirm_password = PasswordField('Confirm Password',
                                     validators=[DataRequired(), EqualTo('password')])
    # 提交按鈕
    submit = SubmitField('Sign Up')

    # 驗證使用者名稱是否已存在
    def validate_username(self, username):
        user = User.query.filter_by(username=username.data).first()
        if user:
            raise ValidationError('That username is taken. Please choose a different one.')

    # 驗證電子郵件是否已存在
    def validate_email(self, email):
        user = User.query.filter_by(email=email.data).first()
        if user:
            raise ValidationError('That email is taken. Please choose a different one.')

```

- 使用者輸入資料驗證，確保格式正確並防止不安全輸入。

第 4 部分：Database

功能：

- 使用 SQLAlchemy 建立資料庫模型。
- 初始化資料庫並管理用戶與貼文資料。

程式碼分析：

models.py - 資料庫模型：

```python
# 定義 User 模型類
class User(db.Model, UserMixin):
    # 用戶 ID，主鍵
    id = db.Column(db.Integer, primary_key=True)
    # 用戶名，唯一且不能為空
    username = db.Column(db.String(20), unique=True, nullable=False)
    # 電子郵件，唯一且不能為空
    email = db.Column(db.String(120), unique=True, nullable=False)
    # 頭像檔案名稱，預設為 'default.jpg'
    image_file = db.Column(db.String(20), nullable=False, default='default.jpg')
    # 密碼雜湊值
    password = db.Column(db.String(60), nullable=False)
    # 與文章的關聯，建立反向關聯 'author'
    posts = db.relationship('Post', backref='author', lazy=True)

    # 產生密碼重設 Token
    def get_reset_token(self, expires_sec=1800):
        s = Serializer(current_app.config['SECRET_KEY'], expires_sec)
        return s.dumps({'user_id': self.id}).decode('utf-8')

    # 驗證密碼重設 Token
    @staticmethod
    def verify_reset_token(token):
        s = Serializer(current_app.config['SECRET_KEY'])
        try:
            user_id = s.loads(token)['user_id']
        except:
            return None
        return User.query.get(user_id)

    # 用於打印用戶資訊
    def __repr__(self):
        return f"User('{self.username}', '{self.email}', '{self.image_file}')"

```

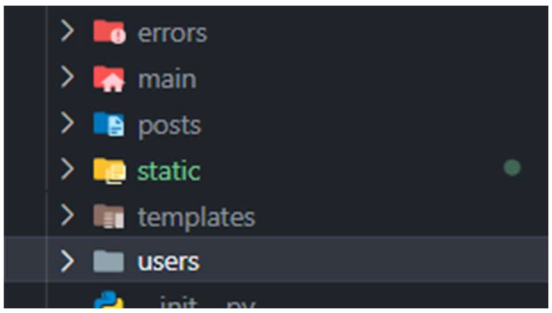
- 定義用戶與貼文模型，並關聯資料表。

第 5 部分：Package Structure

功能：

- 重構應用程式結構，使用 `Blueprint` 提升模組化管理能力。

檔案結構分析：



- 各功能區分為模組，有助於管理與擴展。

第 6 部分：Login Authentication

功能：

- 使用 `Flask-Login` 處理登入與登出。
- 管理存取權限與登入狀態。

程式碼分析：

users/forms.py - login

```python
# 更新帳戶資訊路由
@users.route("/account", methods=['GET', 'POST'])
@login_required
def account():
    form = UpdateAccountForm()
    if form.validate_on_submit():
        if form.picture.data:
            picture_file = save_picture(form.picture.data)
            current_user.image_file = picture_file
        current_user.username = form.username.data
        current_user.email = form.email.data
        db.session.commit()
        flash('Your account has been updated!', 'success')
        return redirect(url_for('users.account'))
    elif request.method == 'GET':
        form.username.data = current_user.username
        form.email.data = current_user.email
    image_file = url_for('static', filename='profile_pics/' + current_user.image_file)
    return render_template('account.html', title='Account',
                           image_file=image_file, form=form)
```

- 限制未登入用戶存取帳號管理頁面，提高安全性。

第 7 部分：User Account and Profile Pic

功能：

- 更新帳號資訊與頭像圖片。

程式碼分析：

users/utils.py -profile

```python
# 儲存上傳的個人圖片

def save_picture(form_picture):
    # 生成隨機的檔案名稱，以避免重複
    random_hex = secrets.token_hex(8)
    _, f_ext = os.path.splitext(form_picture.filename)
    picture_fn = random_hex + f_ext
    picture_path = os.path.join(current_app.root_path, 'static/profile_pics', picture_fn)

    # 調整圖片大小並儲存
    output_size = (125, 125)
    i = Image.open(form_picture)
    i.thumbnail(output_size)
    i.save(picture_path)

    return picture_fn

```

- 確保圖片大小限制與安全處理，支援縮圖功能。

第 8 部分：Posts

功能：

- 新增、編輯與刪除貼文。

程式碼分析：

posts/routes.py

```python
# 建立新文章
@posts.route("/post/new", methods=['GET', 'POST'])
@login_required
def new_post():
    # 初始化表單
    form = PostForm()
    # 驗證表單輸入
    if form.validate_on_submit():
        # 創建新文章並設置作者
        post = Post(title=form.title.data, content=form.content.data, author=current_user)
        db.session.add(post)
        db.session.commit()
        flash('Your post has been created!', 'success')
        # 重定向到首頁
        return redirect(url_for('main.home'))
    # 渲染新文章模板
    return render_template('create_post.html', title='New Post',
                           form=form, legend='New Post')
```

第 9 部分：Pagination

功能：

- 使用 Flask-SQLAlchemy 提供分頁功能。

程式碼分析：

main/routes.py

```python
@main.route("/home")
def home():
    # 取得 URL 查詢參數中的頁碼，預設為第 1 頁，且類型為整數
    page = request.args.get('page', 1, type=int)
    # 根據文章發布日期降序排序，並每頁顯示 5 篇文章
    posts = Post.query.order_by(Post.date_posted.desc()).paginate(page=page, per_page=5)
    # 渲染首頁模板並傳遞文章資料
    return render_template('home.html', posts=posts)

```

- 確保貼文按頁面分組顯示。

第 10 部分：Password Reset Email

功能：

- 提供密碼重設與電子郵件通知功能。

程式碼分析：

users/utils.py

```python
# 傳送密碼重設郵件

def send_reset_email(user):
    # 生成重設密碼的 token
    token = user.get_reset_token()
    msg = Message('Password Reset Request',
                  sender='noreply@demo.com',
                  recipients=[user.email])
    # 設置郵件內容
    msg.body = f'''To reset your password, visit the following link:
{url_for('users.reset_token', token=token, _external=True)}

If you did not make this request then simply ignore this email and no changes will be made.
'''
    # 傳送郵件
    mail.send(msg)
```

第 11 部分：Blueprints

功能：

- 使用藍圖管理功能模組，提高維護與擴展性。

程式碼分析：

users/routes.py

```python
from flaskblog.users.utils import save_picture, send_reset_email

# 建立 'users' Blueprint
users = Blueprint('users', __name__)

```

users/init.py

```python
# 匯入 Blueprint 模組
    from flaskblog.users.routes import users
    from flaskblog.posts.routes import posts
    from flaskblog.main.routes import main
    from flaskblog.errors.handlers import errors

    # 註冊 Blueprint
    app.register_blueprint(users)
    app.register_blueprint(posts)
    app.register_blueprint(main)
    app.register_blueprint(errors)
```

第 12 部分：Error Pages

功能：

- 設計自訂錯誤頁面，提高用戶體驗。

程式碼分析：

Errors/handlers.py

```python
from flask import Blueprint, render_template

# 建立名為 'errors' 的 Blueprint 模組，處理錯誤頁面
errors = Blueprint('errors', __name__)


# 處理 404 錯誤 (找不到頁面)
@errors.app_errorhandler(404)
def error_404(error):
    # 返回自訂的 404 錯誤頁面及 HTTP 狀態碼 404
    return render_template('errors/404.html'), 404


# 處理 403 錯誤 (禁止訪問)
@errors.app_errorhandler(403)
def error_403(error):
    # 返回自訂的 403 錯誤頁面及 HTTP 狀態碼 403
    return render_template('errors/403.html'), 403


# 處理 500 錯誤 (伺服器內部錯誤)
@errors.app_errorhandler(500)
def error_500(error):
    # 返回自訂的 500 錯誤頁面及 HTTP 狀態碼 500
    return render_template('errors/500.html'), 500

```

完整程式在我的 Github

Github：

https://github.com/shengyunn/Flask_lesson